NASnews
March–April 1998 • Volume 3, Number 2

**In this issue...**

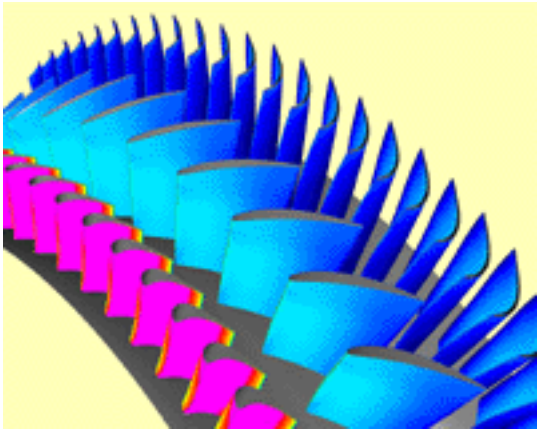


# Team Tests Software to Modernize Legacy CFD Codes

For the last five months, the NAS Systems Division's parallel tools team has been working on a legacy code modernization (LCM) project to study the various means of automating the parallelization of legacy Fortran CFD codes. The hope for the LCM project is that the research done on automation software will bring to light a method for modernizing codes that will replicate the superior results of hand-tuning the code -- without the tremendous investment in time and money.

To The Article...

---



Application of advanced computational methods for the design and analysis of gas turbine systems, based on research performed jointly by Paul Vitt, Mani Subramanian of ASE Technologies, and Mark Turner, David Cherry, Monty Shelton of GE Aircraft Engines.

More information...

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Team Tests Software to Modernize Legacy CFD Codes

**by Ayse Sercan**

**Within This Article...**



For the last five months, the NAS Systems Division's parallel tools team has been working on a legacy code modernization (LCM) project to study the various means of automating the parallelization of legacy Fortran CFD codes. The hope for the LCM project is that the research done on automation software will bring to light a method for modernizing codes that will replicate the superior results of hand-tuning the code -- without the tremendous investment in time and money.

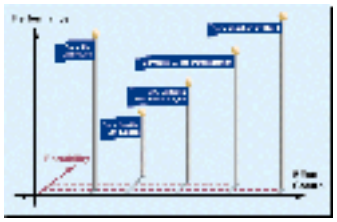
There are a number of reasons why NAS is interested in automating the modernization process. First is the reality that old systems are going away, and researchers need to be able to take advantage of newer architectures. Historically, a new platform has been introduced to the NAS Facility about every three years. For every new platform, legacy code must be translated and optimized, usually at great expense.

Another reason for the interest in automation is tied to the aerospace industry's libraries of legacy codes. Many aerospace companies have not modernized their codes to keep pace with every development in computing architectures because these changes occur so often and the cost of migration is high -- with no guarantee that an investment of time and money will pay off in the long term. By automating at least part of the process, those companies could take advantage of advances in computing technology much sooner, at a lower risk, and at lower cost. This benefits the NAS Facility by ensuring that customers can use new systems, and that those customers get the performance they need.

**Goal is Effective Parallelization**

The usual -- and most effective -- way to parallelize code is by hand, perhaps with the assistance of optimization tools. But this process is time-consuming, takes users away from their primary research, and has to be redone from scratch each time a new architecture is introduced.

The ideal automation software should be able to quickly identify potential problems in the code to be parallelized and then generate code to speed the process, even potentially avoiding some human error that would result from hand-optimizing the code.



Methods of migrating legacy CFD codes

**Looking for 'Swiss Army Knife' Software**

The process of parallelizing legacy code involves first cleaning up the original code to prepare it for translation, then translating the code, and finally doing serial optimizations so that the code runs well even on a single processor. It also involves tuning the code for memory hierarchies that are available on today's machines. Automating that process calls for a wide range of tools -- a sort of Swiss army knife of migration tools.

At this time, the parallel tools team is locating and testing appropriate software for performing the migration tasks. "We'd like to include software that will do code restructuring, including optimizations for both serial and parallel codes. We will also include software that will do performance evaluation," explained Michelle Hribar, a researcher on the parallel tools team.

In addition, the team wants to add visualization software to support source code navigation as well as to provide feedback from the performance analysis. "Nobody's really done a comprehensive comparison of this software before, particularly for large CFD applications," Hribar added, "because it's hard to get hold of these tools, and also because nobody really has the time or resources to test them."

**Taking a Beyond-the-basics Approach**

Beyond the basic tools mentioned above, the team would like to find software that goes one step further and offers different approaches to problems. Hribar noted that "often a researcher will choose one approach to a coding problem and then optimize from there. What we'd like is software that will be able to offer other approaches to the problem -- possibly optimizing the code even more."

**Comparing Available Tools**

The parallel tools team is currently testing software that includes a number of code translators, such as: Computer Aided Parallelization Tools (CAPTools), software that performs source-to-source translation and can generate parallel code with message-passing; The Portland Group's High Performance Fortran, which does the same sort of source-to-source code translation as CAPTools, but can generate either message-passing or shared-memory code; Stanford University Intermediate Format(SUIF) compilers, which translates code for shared-memory systems; and Rice University's D Syste

The team is using these tools to parallelize a test suite of codes that are representative of the legacy codes that NAS is interested in modernizing. This suite includes the sequential versions of the NAS Parallel Benchmarks version 2.3. The parallel codes generated by these tools are migrated to run on the Silicon Graphics/Cray Research Origin2000. Their performance is then compared to that of the hand-coded parallel benchmarks. If all goes as planned, performance results for these tests will be available soon on the parallel tools team's web page.

The team welcomes feedback on these or related tools from anyone who has used them, especially the results of any tests run. Send email to Jerry Yan.

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Storage Group Builds First Phase Of Commodity-based System

**by Ayse Sercan**

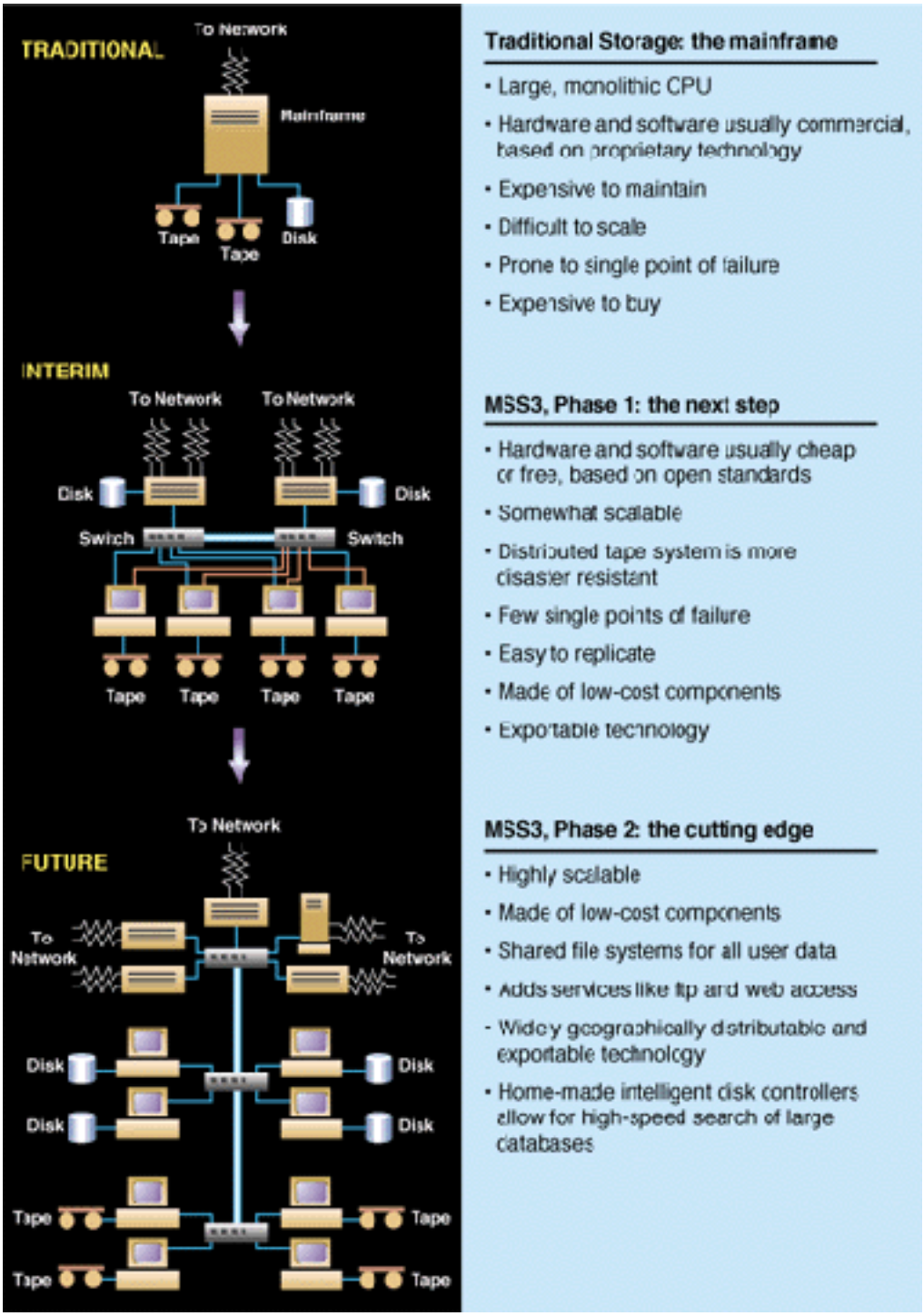
**Within This Article...**



In keeping with the philosophy of "better, faster, cheaper", the NAS storage group is building and testing a system, called Mass Storage Subsystem 3 (MSS3), based on commodity components. "Historically, we have upgraded the entire storage system every three to five years to keep pace with the demands introduced by new high-speed processors," said Harry Waddell, of the storage group. "What we hope to accomplish with this round of change is a shift that will allow us to cheaply and easily add increments of capacity and performance."

About two years ago, the group began working on this commodity storage system to replace the traditional mainframe-plus-peripheral system. MSS3 uses workstations and cheap disks controlled by personal computers (PCs), which essentially act as cheap coprocessors. Relying on NetBSD, a free operating system derived from BSD 4.4 Unix, MSS3 also uses custom software written by the storage group that turns the parts into a functioning system.

The transition to a more cutting-edge system will be eased by the introduction of the first phase of MSS3, which still retains some properties of the current system. This will reduce the upgrade shock somewhat and speed the transition to a faster and better storage system. As an added bonus, it cuts costs by moving away from a big, expensive storage system sooner than could happen with a direct move to the most cutting-edge solution.




## "Virtual Volumes" Offer Stability

One benefit of moving to MSS3 is the use of "virtual volumes" (VV), a collection of files, created in disk cache, which are moved to or from tape en masse. Although tape caching isn't new to the world of storage, the MSS3 strategy uses very aggressive read-and-write cache, Waddell said.

One benefit of using VVs is that they utilize data storage more intelligently. Current D3 tape technology can hold fifty to one hundred

gigabytes (GB) of data -- a volume much larger than is needed for most users -- so a single high-capacity tape can hold several VVs. Virtual volumes speed recovery of data from tape by pre-caching the entire volume when a file is retrieved, so subsequent retrievals are at disk (rather than tape) speed. They also increase the rate at which files can be saved to tape by minimizing tape mounts and positioning.

Because the commodity system is made up of small systems joined by a private internal network, it is also more scalable than most traditional systems. Devices can be added to the network and subtracted as needed, invisible to the user. By using a switched network like HIPPI, performance can be scaled by adding additional links or channels to the network.

MSS3 also will offer more stability than might be expected in a development system. "In NAS storage testbeds, new hardware has proven to be at least as stable as its traditional mainframe counterparts," noted Waddell. And because the commodity system is relatively free of distance limitations (because it connects storage over the network), tape silos and eventually disks can easily be physically distributed, which means improved disaster resistance. According to Waddell, it is possible to build a system that serves multiple sites by using high-speed, wide-area network connections

**New Technology Improves Performance**

The "faster" in "better, faster, cheaper" is not only its improved speed-to-tape, but also reboot and disaster recovery time. Because of the distributed nature of the commodity system, one of the controlling PCs can be rebooted without bringing the whole system down.

This is particularly important (for example) at the NAS Facility, where the fast tape drives are not only very expensive, but can also hang -- taking the rest of their SCSI bus and its devices with them. The expense of these drives may prevent sites from buying the quantity needed to allow the system to operate without them for any length of time. With MSS3, the PC controlling those tape drives can simply be rebooted, without a loss of access to the other data on the system.

**Off-the-shelf Components Mean Freedom**

Instead of using multimillion-dollar mainframes (with expensive service

contracts) and proprietary software and hardware that leaves the division dependent on a single vendor -- and makes technology transfer that much more difficult -- MSS3 primarily uses hardware that can be bought for a few thousand dollars from a computer store and freely distributed software written or modified in house. The first phase of MSS3 is based on Digital Equipment Corp. Alpha 8200s, with Pentium Pro-based PCs acting as coprocessors. The storage group bought off-the-shelf HIPPI switches and PCI network cards from Essential Communication for MSS3's network. With commodity components, the group not only found the best prices, but it is not tied in to one vendor for repairs or upgrades. MSS3 is being tested now, and software components continue to be developed. The system is targeted to begin production mid-year. The current system and MSS3 will run in parallel during a transition period. For more information on NAS storage facilities, send email to John Lekashman.

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Users Get Computing Time Back With Arrival of Second Origin2000 Testbed System

**by Cristy Brickell and Mary Hultquist**

With the February arrival of a new testbed system, researchers on the Computational Aerosciences (CAS) project, can resume work that was delayed when their previous systems were "unplugged" before replacements arrived. Users had been waiting patiently for the new system, which was delayed by federal budget negotiations and NASA budget discussions.

The new 64-processor Silicon Graphics/Cray Research Origin2000, named "Hopper," (16 gigabytes [GB] of memory and 660 GB of RAID-5 disk) uses an 8-processor front-end system that will also be used with Turing, the 64-processor Origin2000 provided by the NAS Systems Division and NASA's Data Assimilation Office (DAO). All three systems are based on the MIPS R10000 processor.

**Within This Article...**



The configurations of both Hopper and Turing are nearly identical, providing similar job performance to users. Tests run with the ARC3D code by Jim Taft were impressive -- with over 6.3 gigaflop-per-second (GFLOP/s) of sustained performance -- equivalent to past performance of ARC3D on a 16-CPU CRAY C90.

## Systems Honor Women in Science

The system is named after Rear AdmiralGrace Murray Hopper, a mathematician and pioneer in data processing who was a legendary

figure among both computer scientists and industry executives. Hopper made several vital contributions to the development of modern computing systems, including helping invent the COBOL programming language and the first practical compiler for computers in the early '60s.

The front-end system, "Evelyn," is named after Evelyn Boyd Granville, one of the first African-American women to earn a Ph.D. in mathematics. In the early '60s, she developed computer programs that were used for trajectory analysis in the Mercury Project (the first U.S. manned mission in space) and in the Apollo project, which sent U.S. astronauts to the moon.

With 8 processors and 2 GB of memory, Evelyn is used for interactive access, including compiling and editing of code. If resources allow, quick debugging runs will be allowed there as well. The cost for both systems was $3.1 million

**Users Lost Computing Resources**

The previous IBM SP2 testbeds located at the NAS Facility and NASA Langley Research Center were obtained through a Cooperative Research Agreement (CRA) with IBM. When the CRA ended last June, use of the systems was extended to provide continued access for CAS researchers. The NAS system shrank from 160 processors to 32 in September, and both systems were decommissioned in late February. Although this extension enabled a transition period to the new platform, the reduction of processors on the NAS system meant that users lost the majority of their computing resources.

**DAO to the Rescue**

During this time, the DAO offered time on their 64-processor SGI/Cray Origin2000 housed at the NAS Facility to the CAS researchers. These user were able to start porting their codes to the new platform to get a jump on the work to come. Users were given almost unlimited access to the system until it was deemed critical to the DAO mission in mid-December.

These extra resources allowed users to prepare to continue their research on Hopper. The change between the distributed memory model of the SP2 and the distributed-shared memory model of the Origin2000 required that some parts of users' codes be rewritten to take advantage of

new machine's architecture. Porting was fairly straightforward, as described in Taft's article.

The systems will also provide a testbed platform for beta software. Both will soon run a beta version of IRIX 6.5, the first "Year 2000 compliant" operating system from SGI. IRIX 6.5 provides several new features, including more robust resource-allocation software, which tries to pin jobs to certain processors and memory locations. This will decrease the possibility of user jobs interfering with one another, ensuring more predictable timings for individual jobs.

**First Operational Period in October**

Because the systems arrived mid-way through the normal operational year, there will be no Announcement of Opportunity during the current period. Former SP2 users can request access by sending email to accounts@nas.nasa.gov. The first operational period will begin on October 1 to coincide with that of the other HPCC program resources. Watch for the FY99 Announcement of Opportunity in an upcoming issue NAS News.

A number of helpful resources are available on the web. These include more detailed information on the Origin2000 systems at the NAS Facility , as well as a quick start and reference guide. For additional assistance with porting and tuning codes, contact the scientific consulting group through NAS User Services by phoning (650) 604-4444 or (1-800) 331-8737, or sending email to nashelp@nas.nasa.gov. The consulting staff is available 7:00 a.m. to 5:30 p.m. Pacific time, weekdays.

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Whitney Update: First Porting Tests are Promising

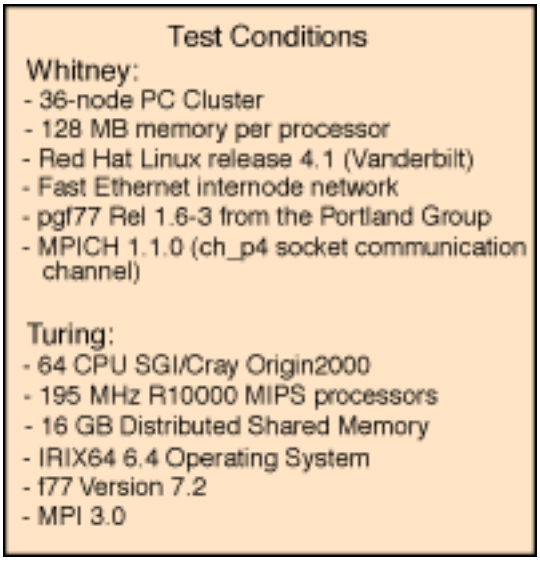
**by Thomas R. Faulkner**

Last fall, members of the NAS parallel systems group reported on the successful installation of the 40-node Whitney prototype. The goal of the Whitney project is to investigate the possibility of creating a production-quality testbed for high-performance computing from a cluster of networked commodity personal computers (PCs).

With the preliminary hardware configuration in place, the Whitney project team is currently focused on collecting, porting, and developing the system software necessary to achieve this goal. To help assess the progress of the project in terms of real-world use, the team decided to port large, production-quality Computational Fluid Dynamics (CFD) applications to the current configuration. This article describes the preliminary porting activity, along with some encouraging initial performance measurements.

**Within This Article...**

## Well-known CFD Codes Selected




The two CFD applications chosen for this test were CFL3D and TLNS3D and, in particular, their distributed MPI variants: CFL3DHP and TLNS3DMPI. CFL3DHP and TLNS3DMPI are large Navier-Stokes flow solvers that use a multigrid approach. This multigrid structure allows for efficient parallelization, with various grid blocks being distributed to separate parallel processes. At the end of each iteration, boundary condition data for interfaces between the grid blocks is shared among the processes that "own" those blocks. This sharing of boundary condition data is performed using the MPI message-passing library.

CFL3DHP and TLNS3DMPI have shown good parallel behavior on several parallel platforms at the NAS Facility, and are widely used throughout the computational aeronautics community. For comparison, similar performance measurements using the same applications and test cases were run on Turing, one of the Silicon Graphics/Cray Research Origin2000 systems at NAS.

## Porting Code Was 'Nontrivial'

Although the codes were successfully ported to the Whitney cluster, they did require a bit of troubleshooting. The main problems encountered with porting both CFL3DHP and TLNS3DMPI to Whitney involved bad code generated by the Fortran compiler. Initial runs showed that CFL3DHP would only run correctly with all optimization disabled, while TLNS3DMPI failed with floating-point errors at all optimization levels.

The CFL3DHP compiler problem was solved relatively quickly. At an intermediate optimization level, this code terminated with an error message that indicated some data problems in one routine. When that routine was compiled with optimization disabled -- while the rest of the

code was compiled with full optimization -- the programs ran correctly.

The problems with porting TLNS3DMPI were a little more difficult to solve. After significant analysis, it was found that two separate routines had to be compiled with optimization disabled. Even then, one of the routines still failed to function correctly. Eventually, some code modifications were made to circumvent the problems that the compiler was having with this particular routine.

In addition to these compiler problems, several smaller issues were addressed in order to successfully port the two codes to the Whitney system. The most significant of these issues involved the large generation environments (multiple interlocking scripts and makefiles) used by these applications to perform configuration manage-

ment and generate executables. Additional complexity was introduced because the two applications use totally different generation environments, so that a fix for a problem in one environment was often completely useless in the other.

Finally, after these nontrivial changes to the code and the generation procedures were applied, the program ran correctly.

**Simple Performance Measurements**

Some simple performance measurements were made using CFL3DHP and TLNS3DMPI on both Whitney and Turing. The results are summarized in Table 1 and Table 2 shown below.

**Table 1**

| | whitney | | turing | |
|---|---|---|---|---|
| Nodes | Run Time | Flop/sec/node | Run Time | Flop/sec/CPU |
| 8 | 11109 | $8.51 \times 10^{+06}$ | 2639 | $35.8 \times 10^{+06}$ |
| 16 | 7460 | $6.34 \times 10^{+06}$ | 1335 | $35.4 \times 10^{+06}$ |

**Table 2**

| | whitney | | turing | |
|---|---|---|---|---|
| Nodes | Run Time | Flop/sec/node | Run Time | Flop/sec/CPU |
| 16 | 2441 | $11.2 \times 10^{+06}$ | 541 | $50.6 \times 10^{+06}$ |
| 32 | 3979 | $3.44 \times 10^{+06}$ | 349 | $39.2 \times 10^{+06}$ |

All run times cited are in wall-clock seconds, as reported by the application. These run times include all program initialization (reading input and grid files) and termination (writing out the solution files). The floating-point performance on Turing was determined by using the madd=ON/OFF procedure.

The Pentium Pro microprocessor (which is the computational heart of each Whitney node) contains a set of performance counters, including a counter for floating-point operations. Unfortunately, the Linux operating system kernel does not provide support for access to these counters. Instead, the floating-point performance on Whitney was estimated from the madd=OFF floating-point operation counts taken from the Turing runs combined with the Whitney wall-clock times.

From these partial scaling results, it is apparent that Turing has a decided advantage over Whitney for message-passing applications such as CFL3DHP and TLNS3DMPI. (The comparatively poor scaling for TLNS3DMPI on Turing turned out to be more of a load-balancing issue than one arising from any significant MPI overhead.) On the other hand, the poor scaling results on Whitney for both CFL3DHP and TLNS3DMPI stem from very large MPI message- passing overhead.

To better illustrate this MPI overhead, let's look at the TLNS3DMPI code run on 32 Whitney nodes. At the end of each run, TLNS3DMPI produces an informative breakdown of total user and system CPU time for each node. For the 32-node run, the average user CPU time per node was 452 seconds and the average system CPU time per node was 483 seconds, for a total of 935 seconds of accumulated CPU time per node. However, the total wall-clock time for this run was 3979 seconds.

This means that only about 12 percent of the overall run time was spent on user CPU cycles and only 23 percent for combined user and system CPU utilization. The remainder of the wall-clock time was spent in the MPI library waiting for grid-block boundary condition data to arrive. These results -- which were not entirely unexpected, as this type of large MPI overhead is fairly typical for fine-grained message-passing applications such as CFL3DHP or TLNS3DMPI -- tend to show up most prominently in cluster configurations such as the Whitney system.

**Cost/Performance Analysis**

Raw speed is not the only important basis for evaluating Whitney's

performance. For instance, it is interesting to compare cost/performance ratios on Whitney and Turing for these two CFD codes (see Table 3 below). In this area, Whitney compares very favorably to large integrated parallel systems.

**Table 3**

| | whitney—$2K/node | | turing—$45K/CPU | | |
|---|---|---|---|---|---|
| Code | Performance | Cost/ Performance | Performance | Cost/ Performance | Turing:Whitney |
| CFL3DHP | $7.43 \times 10^{+06}$ | $269 | $35.6 \times 10^{+06}$ | $1264 | 4.7 |
| TLNS3DMP | $7.32 \times 10^{+06}$ | $273 | $45.0 \times 10^{+06}$ | $1000 | 3.66 |

An approximate cost-per-node estimate for a Whitney node is the purchase price of each PC, which was about $1600, plus another $400 for the network interconnect -- about $2,000 total per node. An approximate cost estimate for a Turing CPU is the total system cost of $2.9 million divided by 64 CPUs -- about $45,000 per CPU. The performance estimates are simple averages of the floating-point performance results for the two-node or CPU counts for each code on each platform. The units are FLOP/sec/node on Whitney and FLOP/sec/CPU on Turing. The unit for the cost/performance ratios is dollars per $10^{+06}$ FLOP/sec. Finally, the last column in Table 3 gives the ratio of the Turing and Whitney cost/performance ratios.

**More Results Forthcoming**

Despite the extremely poor message-passing performance on Whitney, both of these large CFD codes are clearly less expensive to run on Whitney than on Turing (by a factor of about 4-5). At this point, we should emphasize again the preliminary nature of these results. The purpose of these tests was to quickly determine how the Whitney cluster performed under a specific set of "real world" conditions and, as such, it is neither exhaustive nor comprehensive. Additional work to refine and expand the results presented here is already in progress.

---

For continued updates on the Whitney cluster, see the project's web page.

---

*Special thanks to Robert Biedron (CFL3DHP) and Vatsa (TLNS3DMPI), both from NASA Langley Research Center, for providing the source code suitable test problems.*

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Middleware Study Sets Foundation for Building a Distributed Applications Infrastructure

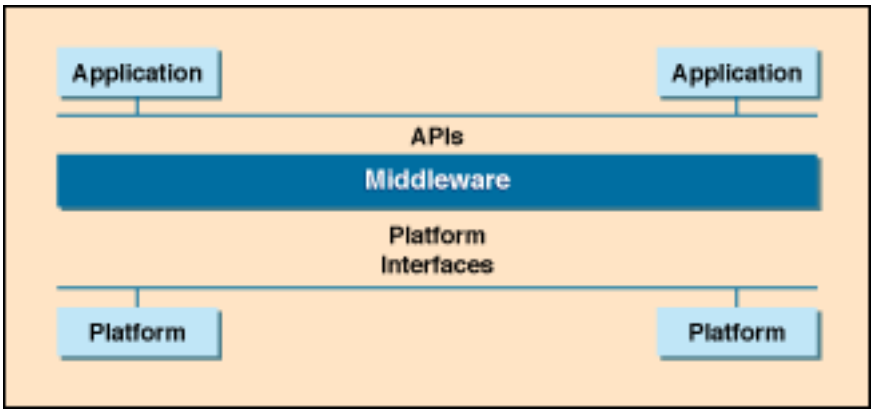
**by Rod A. Fatoohi**

Rod Fatoohi, an associate professor at San Jose State University working with the NAS parallel systems group, presents a solid overview of the state of the art in middleware technology for building a distributed applications infrastructure, such as the Information Power Grid. Four middleware technologies are considered: the Distributed Computing Environment, the Common Object Request Broker Architecture, the Distributed Component Object Model, and Java. Fatoohi gives a brief introduction to middleware concepts, describes the features of each technology studied, then compares the four and makes some conclusions.

Anticipating the need for middleware technologies to support the heterogenous, distributed computing environments that are emerging today, the NAS parallel systems group has conducted a comprehensive study comparing four middleware packages. Findings from this study will be used to make critical choices for the Information Power Grid, and will be useful to NASA and other organizations that are facing the challenges of building efficient and reliable infrastructures for these new environments.

**Within This Article...**

Middleware, like many relatively new, high-level system concepts, is not a well-defined technical term, but generally refers to a set of common services that enable applications and end users to exchange information across networks. These services reside "in the middle" above the operating system (OS) and networking software and below the distributed applications, as shown in the figure below.




The need for middleware stems from computing environments where applications run on various computers, running different operating systems and software tools, and are interconnected by different networks. Middleware integrates these components efficiently and reliably in a distributed heterogeneous environment. Currently, several committees and vendors are working on standards and middleware products to tie together the different applications and computers found in such an environment.

Middleware services have several properties that distinguish them from applications or low-level services; they run on multiple platforms, are used for different applications, are distributed, and -- ideally -- they support standard interfaces and protocols.

**Middleware Pros and Cons**

Middleware has many advantages, including:

- hiding platform software and hardware details from end users
- making application distribution transparent to users
- providing portability, interoperability, flexibility, and scalability
- enabling new applications that use its services
- identifying several important components that can be used and shared across many environments

- helping handle legacy applications

Middleware also has its problems, which include:

- maturity - the technology is not mature enough to use in many mission-critical projects
- standardization - many services use proprietary protocols and application programming interfaces (APIs) and are not based on standards
- cost - middleware adds to the cost of the environment
- performance - middleware may cause a performance degradation in bandwidth and latency
- complexity - many services are complex and not well understood

Middleware technology has become quite important in applications representing a wide range of industries, including: aerospace and defense, banking and finance, telecommunication, and education. It is expected that middleware will also play an important role in many new applications -- such as digital libraries, multimedia, virtual manufacturing, air-traffic control simulation, and the Information Power Grid. These applications need infrastructure support in order for data to move freely among different systems.

**Middleware Offers Many Services**

Early middleware mainly provided communication services. Today's middleware provides many services that can be categorized into three broad areas:

- communication services handle low-level communication between the client and the server, such as the Remote Procedure Call (RPC) mechanism
- management and support services are used by many applications, such as name management, security, failure handling, and memory management
- application-specific services provide assistance for classes of applications, such as SQL (Structured Query Language) database access, transaction processing, and data-replication services

The NAS study focused on middleware that provides most of these services. These middleware technologies are: DCE, CORBA, DCOM, and Java. All of them are considered to be general-purpose, unlike some

technologies that target specific markets, such as database-specific middleware. For detailed information about other types of middleware, see " Middleware for Building Distributed Applications Infrastructure".

**Distributed Computing Environment**

The Distributed Computing Environment (DCE) is a software infrastructure for developing distributed systems. It consists of a set of application programming interfaces and a set of run-time services, which together provide the fundamental functionality required for distributed applications. DCE is based on a simple but flexible Remote Procedure Call (RPC) paradigm. In the middleware arena, DCE is generally considered low level in that it implements primitive fundamental services.

DCE is produced by the Open Group (Cambridge, MA), an integrator specializing in distributed systems and open systems standardization. The Open Group provides a reference DCE implementation and a set of conformance tests. Resellers can purchase the reference implementation and port it to their platforms or develop their own DCE product.

One of DCE's strengths is the number of interoperable implementations available. Vendors that provide DCE products include Cray Research Inc., Digital Equipment Corp., Fujitsu, Hewlett Packard, Hitachi, IBM, NEC, Siemens Nixdorf, Sony, Silicon Graphics Inc., and Tandem Computers, among others.

DCE consists of a set of software packages layered on top of an operating system. In addition to its RPC-based communication services, it provides a set of basic (core) services that includes: a naming service, Cell Directory Service; a security and authentication service; and a time synchronization service, Distributed Time Service. In addition, there are a few application-level services (or application development aids) associated with DCE. Two well-known services are Distributed File Service (DFS) and Encina, a transaction processing support package. Security and CDS are strong components of DCE. DFS provides many attractive features as a distributed filesystem. Encina is an attractive transaction processing monitor that takes advantage of DCE's underlying core services.

DCE uses an Interface Definition Language (IDL) to specify procedure interfaces. DCE IDL is similar to other interface languages, including

CORBA IDL, but it is unique (there is no standard for IDLs). Currently, only the C language is directly supported by DCE IDL.

Despite DCE's positive features, it has many problems, including complexity and lack of object support in its architecture. For information about DCE and the NAS Facility's experience with it, see the NAS technical report cited above, or send email to David McNab in the parallel systems group.

**Common Object Request Broker Architecture**

The Common Object Request Broker Architecture (CORBA) is a standard for transparent communication between application objects. The CORBA specification is being developed by the Object Management Group (OMG), a non-profit industry consortium of over 700 software vendors and users involved in the development of object technology for distributed computing systems. The OMG does not produce any software; rather, it provides specifications that come from OMG members who respond to requests for proposals.

In 1990, the OMG introduced a reference architecture for object-oriented applications called the Object Management Architecture (OMA). The latest revision was introduced in 1997. The OMA is mainly composed of an object model and a reference model. The OMA object model defines how objects are specified in a distributed environment. In this client/server model, the servers are objects that provide services to clients, and the clients obtain services by invoking operations on server objects.

The OMA reference model defines the OMA components, their interfaces, and the interactions between them. These components are: Object Request Broker (ORB), Object Services, Common Facilities, Domain Interfaces, and Application Interfaces.

The ORB enables clients and objects to communicate in a heterogeneous distributed environment. It provides the communication service within the middleware. The Object Services provide core services for using and implementing objects. The OMG has specified the following services: Naming, Event, Life Cycle, Persistent Object, Transaction, Concurrency Control, Relationship, Externalization, Query, Licensing, Property, Time, Security, and Object Trader.

The Common Facilities are interfaces for horizontal end-user-oriented applications that can be used by many application domains, such as user interface and information, system, and task management. Domain interfaces are aimed at specific application domains such as manufacTuring, finance, and health. The application interfaces are nonstandardized interfaces specific to a certain application.

The CORBA 1.1 specification, introduced in 1991, describes the interfaces and services that ORBs must have; in effect, CORBA is basically the technology adopted for ORBs. CORBA provides a "clean model" -- that is, the interface of an object and its underlying implementation are separated. Clients do not need to know how or where servers are implemented. Server objects are visible only through interfaces and object references. The specification includes an IDL for describing interfaces. IDL mappings to several languages (such as C, C++, Java, Smalltalk, Ada, and COBOL) have been specified and provided by many vendors.

Currently there are at least a dozen ORBs from different vendors in the commercial market. All provide similar basic services but differ in the number of platforms supported and services provided. Most of them are available on Windows 95, Windows NT, and many Unix platforms. The list of ORBs includes: Orbix, IONA Technologies; VisiBroker, Visigenic; ObjectBroker, BEA Systems, Inc.; ORBplus, HP; and Component Broker Connector, IBM.

CORBA is used in many applications representing different industries. Among them: aerospace and defense, banking and finance, manufacturing, health care, telecommunications, petroleum, transportation, insurance, and education. These applications range from small prototypes and experiments to mission-critical projects.

**Distributed Component Object Model**

Distributed COM is an extension of the Component Object Model (COM), developed by Microsoft Corp. as an object-based programming model for developing and deploying software components. DCOM (previously called Network OLE) adds distributed support to COM to work across a network. The addition includes communication with remote objects, location transparency, and an interface to distributed security services.

COM is the basis of many Microsoft object-based technologies such as ActiveX and Object Linking and Embedding (OLE). ActiveX is now being promoted by Microsoft as a complete environment for components, distributed objects, and Web-related technologies. In essence, ActiveX is Microsoft's label for a wide range of COM-based technologies. (This set of technologies was formerly called OLE; now OLE refers to compound documents only.)

COM-based technologies are directly supported by Microsoft on MS Windows operating systems. Other software vendors, including Software AG and DEC, are currently porting some of these technologies to other platforms, such as Unix. COM has been part of MS Windows for some time. It separates object interfaces from their implementations, similarly to CORBA. Each COM object is an instance of a COM class. One or more COM classes are housed in a COM server, which provides the necessary structure around an object to make it available to clients.

There are three kinds of COM servers: in-process servers execute in the same process as their clients; local servers execute in a separate process from their clients but on the same machine; and remote servers execute in a separate process on a different machine and use DCOM to communicate with remote servers.

COM defines a binary call standard for interfaces. It also defines a language for specifying interfaces called Microsoft Interface Definition Language (MIDL). An MIDL compiler is provided to generate client proxies and server stubs in C or C++ from an interface definition, similar to client stubs and server skeletons in DCE and CORBA.

The DCOM extension enables COM processes to run on different machines. DCOM uses an extension of the DCE RPC for interactions between DCOM objects called Object RPC, or ORPC. The main difference between these two is that when the client asks the registry for the server location, the DCOM registry points to an IP address, and COM points to a location on the local machine.

Both COM and DCOM provide a number of facilities that are similar to CORBA services (but are not called services because they are built into the COM library). These facilities provide functions such as naming, security, transaction, persistence, life cycle, versioning, event, and data access services.

DCOM is integrated into Windows NT 4.0 and can be downloaded free of charge for Windows 95. Software AG has ported DCOM to many Unix platforms, such as Sun Solaris 2.5 and Digital Unix 4.0., and is in the process of porting it to other platforms.

**Java**

Java, developed by Sun Microsystems Inc., is a label for a broad range of object-based technologies for the web and distributed applications. It is an object-oriented programming language as well as a computing platform for developing and running distributed applications. The Java Platform consists mainly of the Java Virtual Machine (JVM) and Java API, which includes Java core classes. The Java Platform sits on top of many other platforms and compiles to bytecodes -- machine-independent instructions for the JVM.

The Java Platform is currently embedded in browsers such as Netscape Navigator, and runs on many other platforms such as MS Windows, Unix, IBM MVS, and network operating systems. The Java API, a multiplatform standard interface, specifies a set of interfaces for a wide range of applications. There are two types of Java API: the Java Base API, which defines the core features for developing and deploying Java programs, and the Java Standard Extension API, which extends the capabilities of the core API.

Java programs are categorized as applets, applications, or JavaBeans. Applets are small or modular programs that require a Java-compatible browser to run. Java applications, on the other hand, are stand-alone programs that do not require a browser. And JavaBeans are reusable software components that are platform independent and allow dynamic interactions with other components. Unlike an applet, a JavaBean can interact with other components over the network and can run in containers other than a browser.

Distributed objects, such as JavaBeans, can interact with other applications across a network using the Java Remote Method Invocation (RMI), which enables objects in one JVM to invoke methods on objects in a remote JVM. The invocation methods on both local and remote objects use the same syntax. An RMI compiler (RMIC) generates stubs and skeletons, as in CORBA and DCOM. Recently, Sun announced that Java RMI will support CORBA IIOP as a transport protocol in addition to its native transport protocol.

Java provides many services -- though most are not well defined -- that are similar to CORBA services. Some have been developed along with other Java technologies such as JavaBeans and Java Platform. Among these services are security, Java DataBase Connectivity, Java Naming and Directory Interface, Reflection and Introspection, Event, Persistence, and Java Transaction Service.

**Similarities and Differences**

The four middleware technologies described here provide somewhat similar functionality; however, there are some key differences -- including maturity, ownership, standardization, platform support, and services provided.

Although DCE and CORBA have been around for several years, in general the former is more mature and has a more complete set of core services than the others. But, it suffers from the lack of object support in its architecture. And although CORBA has a *richer* set of services than the others, many of those have been approved just recently and will take time to be implemented. In addition, many CORBA facilities and some services are still being debated by the OMG. It is expected to take about two more years for CORBA to have a complete, mature, and fully implemented set of services. That should *not* dissuade software developers from using CORBA now, as it is the only general-purpose middleware based on object-oriented technology.

Both DCOM and Java are owned, specified, and developed primarily by their vendors, Microsoft and Sun, respectively. The advantages and disadvantages of vendor-specified products, compared to standards committee specifications, are well known and understood in the industry.

Among the four technologies, only DCOM is not widely supported on multiple platforms. It runs mainly on MS Windows and on a few Unix systems through third-party vendors. DCE, CORBA, and Java are widely supported on many platforms.

**Middleware Progress Continues**

Middleware will become increasingly sophisticated because it allows users to access remote services as if they were local. Vendors and users are increasingly dependent on standards, as they want to ensure that

their investment in developing and integrating middleware services pays off. Many committees are working on introducing standards into industrial products, and many vendors are pushing to make their products *de facto* standards.

Because of the constant changes in middleware technology -- which are hard to stay on top of without vigilance -- this market may take some time to stabilize. In the meantime, each organization should choose its middleware based on current and future needs. Because middleware is an infrastructure, an initial error in choice will not be so easy to correct.

The NAS parallel systems group plans to do more work in analyzing these four technologies to provide a comprehensive view of their functionality and limitations. One area of interest is in integrating legacy applications, such as CFD applications, within a CORBA environment.

---

For more in-depth information on this study, send email to the author at **fatoohi@nas.nasa.gov**.

---

*Rod A. Fatoohi is an associate professor in computer engineering at San Jose State University and a research consultant with the NAS Systems Division. His research interests include distributed objects, parallel and distributed systems, computer networking, and performance analysis. He holds a Ph.D. in electrical engineering.*

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# *High-speed Processor Techniques*

More than a year-and-a-half after Cray Research discontinued support for its Fortran 77 compiler, the NAS and Aeronautics Consolidated Supercomputing Facilities removed the compiler from the Cray supercomputers. In these two pieces, members of the NAS scientific consulting group discuss the ramifications of this change, its impact on the user community, and some related information on new utilities.

---

# Cray Fortran 77 Compiler 'Bytes' the Dust

**by Johnny Chang, Art Lazanoff, and George Myers.**

On March 2, the Cray Fortran 77 compiling system (CF77) was removed from the NAS and ACSF Cray vector processors, VonNeumann, Eagle, and Newton. Although Cray Research stopped supporting CF77 in August 1996, replacing it with their Fortran 90 compiler (CF90), the NAS Facility continued to run the old compiler until CF90 was more mature, and to give users substantial time to make the transition.

**Within This Article...**

The decision to finally remove CF77 was based on several factors. First, Cray will not provide the compiling system much longer -- or guarantee that it will work with newer versions of the operating system (an upgrade to UNICOS 10.0 is anticipated in late March or early April). Second, the Fortran language standard (ANSI X3.198-199X) encompasses the Fortran 77 dialect with some minor changes, which means that a user's Fortran 77 program should compile and run correctly under the new language. Third, the NAS scientific consultants have run tests which show that CF90 produces code that performs as well as -- and in many cases better than -- code produced by CF77. Finally, the CF90 compiler is supported and will continue to be improved over time.

**The Price of Change**

Change does not come without a price. Because the new Fortran 90 language is much larger than its predecessor, the compiler is also larger and somewhat slower. For the most part, CF90 produces faster executable code. Cray referred to CF77 as a compiling system that included FPP, the Fortran Preprocessor and FMP, the Fortran Midprocessor, as stand-alone programs that could also be invoked by the *cf77* command. These functions are now integrated into CF90. As a result, users will need to update or delete some makefile options and user directives.

In addition, because the CF90 compiler is still maturing, some codes will uncover bugs. In fact, several have already been found during the conversion process.

**The Good News**

Despite these issues, the benefits *do* outweigh the price. The new language includes valuable programming features such as dynamic memory allocation, more concise notation for precision, array allocation and usage, and modules for consistent data- and code-sharing among subprograms. These features contribute to decreased code development time, greater robustness of a program, and improved run-time flexibility.

The CF90 compiler is invoked through the *f90* command. The table below shows some of the common flags used for *cf77* and their *f90* counterparts. For a more complete comparison of these flags, see "Compiling Code on vn" and look for the section titled "A Comparison

of Options of f90 vs. cf77."

| cf77 Flags | f90 Flags | Explanation |
|---|---|---|
| -Zv | -Oscalar3, vector 3 | maximum vectorization but no autotas |
| -Zp | -Oscalar3, vector, 3, task 3 | maximum vectorization with autotaskir |
| -Wf "-a static" | -ev | allocates variables to static storage |
| -Wf "-e m" | -rm (-r2 recommended | loopmark listing |

The default flag used by the CF90 compiler is the -O2 level of optimization. More information about *f90* options is available from the man pages (type "man f90").

**Optimization Tips on Web**

The NAS scientific consultants maintain web pages for VonNeumann, Eagle, and Newton that identify much of what has been learned in their experience with Fortran 90. (See the section titled "Tips for Optimization" at the following locations, respectively:

Contact NAS User Services for any assistance, particularly if your code doesn't perform as well when compiled by CF90.

---

# 'Resurrected' UNICOS Fortran Utilities Being Beta Tested

**by**

**Clayton J. Guest**

In August 1996 when Fortran 90 became available on the Cray parallel vector processor supercomputers, the UNICOS utilities *fsplit* and *fmgen* were unusable with the Fortran 90 program files. Recently, Silicon Graphics/Cray Research released for beta testing two new utilities, called *ftnsplit* and *ftnmgen*, which provide the functions of the old utilities for Fortran 90 files.

The *ftnsplit* utility splits named Fortran files into separate files, with one unit per file, and lists the resultant files. Each unit includes the following program segments: blockdata, function, main program, module, and subroutine. The *ftnmgen* utility splits out all Fortran subroutines from files by using the *ftnsplit* utility, and produces a makefile that can be used to compile and load a program. By using an option flag, *ftnmgen* can create a file-based makefile, the creation of which does not use *ftnsplit* to split the files. The *ftnmgen* utility inserts dependency rules for include files and module usage.

**Get Help From Man Pages, User Services**

Both *ftnsplit* and *ftnmgen* have man pages on the Cray parallel vector systems. If you have difficulty viewing these pages, then issue the command "module switch craytools craytools_new" to UNICOS to get access.

Because these utilities are still being beta tested, please report any problems or difficulties to NAS User Services.

Contact NAS User Services 24 hours daily, 7 days a week at: (650) 604-4444 or (800) 331-8737, or by email.

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# New Research Group Forges Ahead

**by Jill Dunbar**

**Within This Article...**



Just six months after the Information Technology (IT) modeling and simulation group was established in the NAS Systems Division by acting chief Paul Kutler, researchers have made progress on a number of innovative projects. The group has also begun collaborating with industry, universities, and national laboratories to further NASA research goals.

Led by research scientist Subhash Saini, the group's broad charter is to model and simulate emerging computer- and communications-related technologies relevant to NASA missions. These technologies include quantum devices, molecular electronics, and optical interconnects, as well as modeling performance of existing and future information systems.

The group, a mix of government and contractor staff, is comprised of two teams that specialize in semiconductor device modeling -- which focuses on the quantum effects in meso- and macroscopic structures -- and computational molecular nanotechnology, which involves working with experimentalists to design and manipulate atomically precise components.

## Research Will Pay Off in Long Term

The two teams are pursuing research in a number of areas, including:

- Computer simulations for testing the feasibility of hydrogen storage

in carbon nanotubes for lightweight fuel storage on NASA missions;

- Computational quantum optoelectronic devices, which deal with modeling and simulating the interaction between semiconductor nanostructures and laser light for transmitting and processing information. Such devices include vertical cavity surface-emitted lasers, modulators, and detectors;
- Developing and testing large-scale simulations of the mechanical and electrical properties of carbon nanotubes for use in creating next-generation electronic, mechanical, and sensing devices.
- DNA computing (also called molecular computing): a new approach to massively parallel computation, in which calculations are done by synthesizing DNA sequences.

Although much of this work has potential for application in the long term, some of it will be applied to actual problems in the near future. For instance, researcher Cun-Zheng Ning predicts that optical electronic devices will likely replace or work in combination with standard copper wire technology within a few years, to meet the demand for very-high-speed communication.

**Collaborating With Outside Organizations**

Last August, over 200 participants -- about 90 of which were from industry -- attended the NAS-sponsored Second Device Modeling Workshop. As an outcome of the workshop, several joint projects were begun.

In one such effort, M.P. Anantram and Jie Han, both at NAS, provide carbon nanotube models to T.R. Govindan at Pennsylvania State University, for his work in large-scale simulation of atomic structures. Han is also collaborating with Hongjie Dai at Stanford University. Using a scanning probe microscopy technique, their work has demonstrated the smallest and strongest writing instrument in existence -- "nanopencils" consisting of 10 atoms arrayed in a pencil-like shape, which etch silicon surfaces. These nanopencils will someday be used to read and write information for very-high-density disk storage.

**Get More Information**

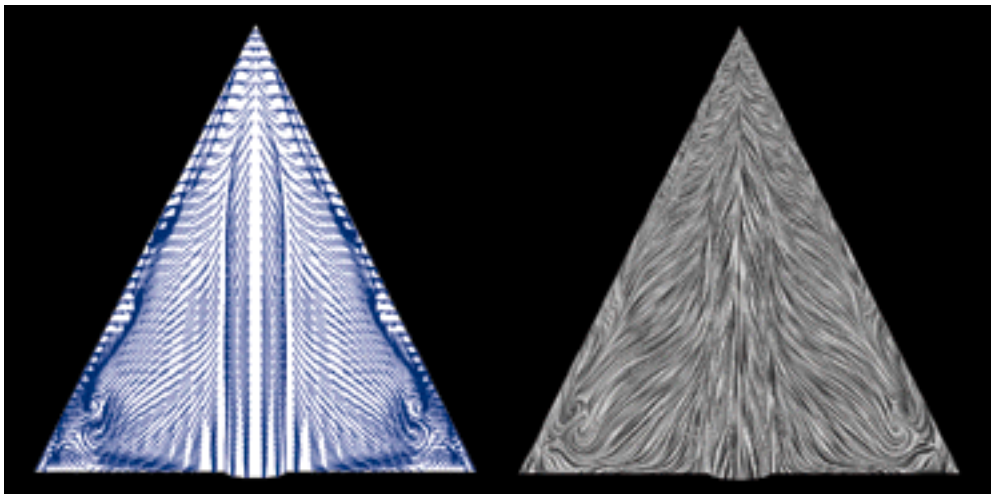
A wealth of information on Ames device modeling and simulation activities is available online. More information on these and other related projects will be detailed in future issues of NAS News.

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Researchers Present Comparisons of Surface Flows at AIAA '98



NAS researchers presented comparisons of two visualization techniques at the annual AIAA Aerospace Science Meeting and Exhibit held in January. This graphic shows surface flows on a delta wing computed using both streamlines (left) and the Line Integral Convolution (LIC) technique (right). The flow lines are more continuous using the LIC technique; the streamlines depict the flow reattachment lines, but do not reveal flow separation lines clearly. The LIC technique reveals both separation and reattachment lines, and the vortex structures are more evident in the LIC technique. Dataset by Neal Chaderjian, NASA Ames Applied Computational Aerodynamics Branch; graphics by David Kao.

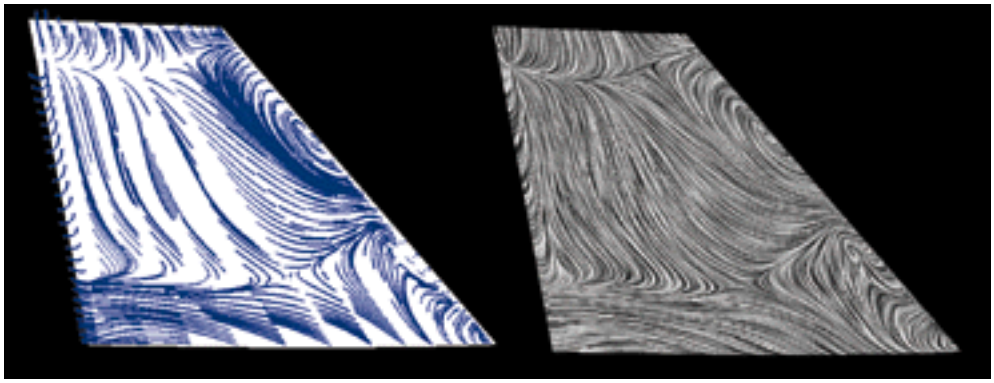
Recently, NAS Systems Division researchers David Kao and Han-Wei Shen presented the results of their paper, "Numerical Surface Flow Visualization" (AIAA-98-0076) at the 36th American Institute of Aeronautics and Astronautics (AIAA) Aerospace Sciences Meeting and Exhibit, held in Reno, NV, January 12-15.

The paper compared the results of two methods of visualizing CFD surface flows: streamlines, in which a particle trace is computed from

each grid point, and the Line Integral Convolution (LIC), a texture synthesis technique that closely resembles wind tunnel results using surface oil flows. For unsteady flow simulations, streaklines (time-dependent particle traces) were compared with a new texture synthesis technique called Unsteady Flow Line Integral Convolution (UFLIC).

Kao and Shen were able to do several comparisons of surface flows represented by both streamlines and the LIC technique. They found that streamlines used to depict surface flows are discontinuous in general and the quality of the surface flow pattern is highly dependent on the placement of the streamlines.

---



Surface flows computed using the tail of an F/A-18 fighter jet. The saddle points at the upper left and the lower center of the tail are very clear in the Line Integral Convolution (LIC) technique (right). Due to the grid resolution, the saddle points are not depicted clearly with the streamlines (left). Dataset by Ken Gee, Ames Applied Computational Aerodynamics Branch; graphic by David Kao.

---

In contrast, the LIC technique clearly depicts surface flows that closely resemble surface oil flows. In addition, surface flows near regions of vortex structures and saddle points (as shown above) are shown better using the LIC technique than using the streamline technique.

For unsteady flow simulations, Kao and Shen compared two instantaneous surface flow visualization techniques (streamlines and LIC) with two time-dependent flow visualization techniques (streaklines and UFLIC). When the results were animated and run side-by-side, it was apparent that UFLIC accurately reveals the dynamic behavior of unsteady surface flows, as shown in the graphic above.

Future plans include further surface flow comparisons with more datasets -- both CFD and experimental data -- and more comparisons of surface flows in unsteady flow fields.

For more information about numerical surface flow visualization, send email to Kao or to Shen.

- More about UFLIC

NASnews
▶ Main Menu
March–April 1998 • Volume 3, Number 2

**In this issue...**



# Credits

**Executive Editor:** Thomas Lasinski

**Editor:** Jill Dunbar

**Staff Writer:** Ayse Sercan

**Contributing Writers:** Cristy Brickell, Johnny Chang, Rod A. Fatoohi, Thomas R. Faulkner, Clayton J. Guest, Mary Hultquist, George Myers

**Image Coordinator**: Chris Gong **Online Page Layout and Graphics:** Joel Antipuesto, Chris Gong, Eunah Choi, Rosemary Wadden

**Other Contributors:** Richard Anderson, Jeffey Becker, Archie deGuzman, James Donald, Gail Felchle, John Hardman, Ed Hook, Michelle Hribar, Randy Kaemmerer, David Kao, John Lekashman, David McNab, Terry Nelson, Marcia Redmond, Karl Schilke, Jerry Yan

**Editorial Board**:
Cristy Brickell, Jill Dunbar, Chris Gong, Thomas Lasinski, Nateri Madavan, Patrick Moran, George Myers, Ayse Sercan, Harry Waddell